User Guide

Installing Linux on the F75P

F75P – 3U CompactPCI® PlusIO Safe Railway Computer

About this Document

This user guide is intended only for system developers and integrators, it is not intended for end users.

It deals with the installation of Linux on the F75P I/O Processor (IOP) and Control Processors (CP1 and CP2). The document also describes a demo application for F75P that is available for download on MEN's website.

This user guide is not intended to be used for safety-critical applications.

You can find all Safety Related Application Conditions in the F75P user manual.

In any case we strongly recommend that you make yourself familiar with the F75P architecture before you start working with software.

If you do not have access to the F75P user manual yet, please contact MEN for more information.

History

Issue	Comments	Date
E1	First issue	2014-10-13

Conventions

Indicates important information or warnings concerning proper functionality of the product described in this document.

The globe icon indicates a hyperlink that links directly to the Internet, where the latest updated information is available.
When no globe icon is present, the hyperlink links to specific elements and information within this document.

italics	Folder, file and function names are printed in *italics*.
bold	**Bold** type is used for emphasis.
`mono`	A `monospaced` font type is used for hexadecimal numbers, listings, C function descriptions or wherever appropriate. Hexadecimal numbers are preceded by `"0x"`.
comment	Comments embedded into coding examples are shown in green text.
IRQ# /IRQ	Signal names followed by a hashtag "#" or preceded by a forward slash "/" indicate that this signal is either active low or that it becomes active at a falling edge.
in/out	Signal directions in signal mnemonics tables generally refer to the corresponding board or component, "in" meaning "to the board or component", "out" meaning "from it the board or component".
	Blue vertical lines in the outer margin indicate sections where changes have been made to this version of the document.

Legal Information

Changes

MEN Mikro Elektronik GmbH ("MEN") reserves the right to make changes without further notice to any products herein.

Warranty, Guarantee, Liability

MEN makes no warranty, representation or guarantee of any kind regarding the suitability of its products for any particular purpose, nor does MEN assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages. TO THE EXTENT APPLICABLE, SPECIFICALLY EXCLUDED ARE ANY IMPLIED WARRANTIES ARISING BY OPERATION OF LAW, CUSTOM OR USAGE, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR USE. In no event shall MEN be liable for more than the contract price for the products in question. If buyer does not notify MEN in writing within the foregoing warranty period, MEN shall have no liability or obligation to buyer hereunder.

The publication is provided on the terms and understanding that:

1. MEN is not responsible for the results of any actions taken on the basis of information in the publication, nor for any error in or omission from the publication; and

2. MEN is not engaged in rendering technical or other advice or services.

MEN expressly disclaims all and any liability and responsibility to any person, whether a reader of the publication or not, in respect of anything, and of the consequences of anything, done or omitted to be done by any such person in reliance, whether wholly or partially, on the whole or any part of the contents of the publication.

Conditions for Use, Field of Application

The correct function of MEN products in mission-critical and life-critical applications is limited to the environmental specification given for each product in the technical user manual. The correct function of MEN products under extended environmental conditions is limited to the individual requirement specification and subsequent validation documents for each product for the applicable use case and has to be agreed upon in writing by MEN and the customer. Should the customer purchase or use MEN products for any unintended or unauthorized application, the customer shall indemnify and hold MEN and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim or personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that MEN was negligent regarding the design or manufacture of the part. In no case is MEN liable for the correct function of the technical installation where MEN products are a part of.

Trademarks

All products or services mentioned in this publication are identified by the trademarks, service marks, or product names as designated by the companies which market those products. The trademarks and registered trademarks are held by the companies producing them. Inquiries concerning such trademarks should be made directly to those companies.

Conformity

MEN products are no ready-made products for end users. They are tested according to the standards given in the Technical Data and thus enable you to achieve certification of the product according to the standards applicable in your field of application.

RoHS

Since July 1, 2006 all MEN standard products comply with RoHS legislation.

Since January 2005 the SMD and manual soldering processes at MEN have already been completely lead-free. Between June 2004 and June 30, 2006 MEN's selected component suppliers have changed delivery to RoHS-compliant parts. During this period any change and status was traceable through the MEN ERP system and the boards gradually became RoHS-compliant.

WEEE Application

The WEEE directive does not apply to fixed industrial plants and tools. The compliance is the responsibility of the company which puts the product on the market, as defined in the directive; components and sub-assemblies are not subject to product compliance.

In other words: Since MEN does not deliver ready-made products to end users, the WEEE directive is not applicable for MEN. Users are nevertheless recommended to properly recycle all electronic boards which have passed their life cycle.

Nevertheless, MEN is registered as a manufacturer in Germany. The registration number can be provided on request.

Copyright © 2014 MEN Mikro Elektronik GmbH. All rights reserved.

Germany
MEN Mikro Elektronik GmbH
Neuwieder Straße 3-7
90411 Nuremberg
Phone +49-911-99 33 5-0
Fax +49-911-99 33 5-901
E-mail info@men.de
www.men.de

France
MEN Mikro Elektronik SAS
18, rue René Cassin
ZA de la Châtelaine
74240 Gaillard
Phone +33 (0) 450-955-312
Fax +33 (0) 450-955-211
E-mail info@men-france.fr
www.men-france.fr

USA
MEN Micro Inc.
860 Penllyn Blue Bell Pike
Blue Bell, PA 19422
Phone (215) 542-9575
Fax (215) 542-9577
E-mail sales@menmicro.com
www.menmicro.com

Contents

1 General 8
1.1 Running Linux on F75P 8
1.2 Installation of Ubuntu Trusty using debootstrap 9

2 Installing Linux 10
2.1 Preparing a Mass Storage Device 10
2.2 Configuring the IOP 10
2.2.1 Root File System 10
2.2.2 Additional Software Packages 10
2.2.3 Building a Linux Kernel on the Development System 11
2.2.4 Configuring the Kernel 12
2.2.5 Modifying Drivers 13
2.2.6 Building and Installing the Kernel 14
2.2.7 Installing the GRUB Boot Loader 15
2.3 Installing MDIS 16
2.3.1 Preparing for Installation 16
2.3.2 Configuring MDIS using MDIS Wizard 16
2.3.3 Building and Installing MDIS 24
2.3.4 Creating initrd 25
2.3.5 Configuring the Network 25
2.3.6 Creating a New User and Setting the Root Password 25
2.3.7 Installing a TFTP Server 26
2.3.8 Installing a DHCP Server 26
2.3.9 Installing an NFS Server 26
2.3.10 Additional Modules Loading on Boot 27
2.3.11 Instantiating the System RTC 27
2.4 Configuring CP1 and CP2 27
2.4.1 CP Root File System 27
2.4.2 Additional Software for the CP 28
2.4.3 Creating Configuration Files for PXE Boot 28
2.4.4 Downloading the PXE Linux Boot File 28
2.4.5 Creating initrd for Each of the CPs 29
2.4.6 Setting the Host Name for Each CP 29
2.4.7 Setting the Password for Root in CP1 and CP2 30
2.4.8 Configuring the Watchdog Trigger using a Script 30
2.5 Using the Installation on F75P 32

3 Using F75P Functions under Linux 33
3.1 Graphical User Interface (GUI) 33
3.2 Real-Time Clock (RTC) 33
3.3 Board Management Controller (BMC) 34
3.4 Board Information EEPROM 35

4 Demo Application 36
4.1 About the Application 36
4.1.1 General Hardware and Software Requirements 36
4.2 Installing the Demo Application 37
4.3 Using the Demo Application 38
4.3.1 Start 38
4.3.2 General Behavior of the Application 38
4.3.3 Functions 38
5 Appendix 40
5.1 Literature and Downloads 40
5.1.1 CPU Board and BIOS 40
5.1.2 Ubuntu Linux 40
5.1.3 Embedded Linux 40
5.1.4 Linux Kernel Configuration 40
5.1.5 Linux Device Drivers 40
5.1.6 Linux System Programming 41
5.1.7 General 41

1 General

1.1 Running Linux on F75P

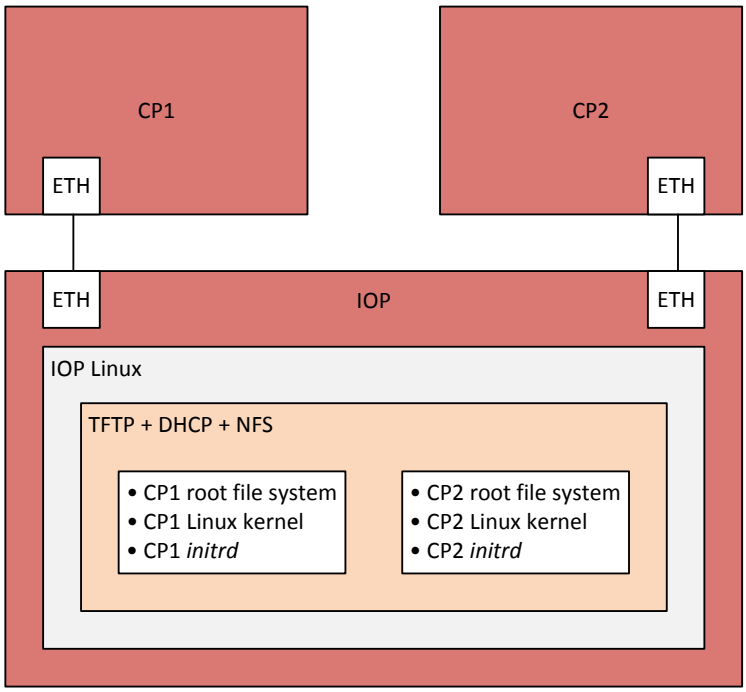
All processors of F75P (Control Processors CP1 and CP2, and the I/O Processor IOP) support Linux. You will need a boot image for every processor in the F75P system.

Note: This document assumes that Linux is to run on all three processors. Of course, the three processors also support other operating system constellations. However, the basic principle is always the same.

As the CPs exclusively employ network boot using the board's internal Ethernet connection to load their boot images, they need a DHCP/TFTP server running on the IOP.

The basic concept of running Linux on F75P is to start up a Linux on the IOP and provide the kernel, *initrd* (initial ramdisk) and a root file system for each of the CPs. The CPs wait until the TFTP and DHCP servers are running on the IOP and then boot the files provided by the IOP. The figure below shows the files and servers running on each system.

Figure 1. *Linux on F75P block diagram*



Please note that MEN does **not** provide a dedicated Linux BSP for F75P but provides the necessary OEM software support through the MEN MDIS5 system package for Linux and through dedicated driver software.

1.2 Installation of Ubuntu Trusty using *debootstrap*

You can generally use different Linux distributions with F75P. As an example, this document describes installation of Ubuntu Trusty with kernel 3.15.6 using *debootstrap* for the root file system. Ubuntu Saucy 32-bit was used on the development machine (Ubuntu 13.10). If you use a different distribution/version, some commands, e.g., *debootstrap* may vary.

It is not possible to install the Linux system directly on the F75P because each of the CPs has a watchdog that will signal an error if it is not triggered through software. The timeout for the initial trigger is too short for installing an operating system.

To install Linux, you need to prepare the operating system on a SATA mass storage device that is directly connected to the development machine (e.g., using a SATA port or a USB-to-SATA adapter).

Requirements:

- Linux development system or Linux Virtual Guest system
- Internet connection on the development system
- USB-to-SATA adapter (if necessary)
- MEN driver software:
  - 13MD05-90 MDIS5 System Package for Linux
  - 13Z077-91 Linux Ethernet driver for F75P

MEN driver software is available for download on the F75P product pages, under "Downloads":
www.men.de/products/02F075P.html

Note: The 13MM02-90 Linux driver for the RX8581 real-time clock is available for Linux kernels up to 3.14. The driver is already included in the upstream Linux kernel from kernel release 3.14. For the installation described in this user guide, you do not need the additional RTC driver.

2 Installing Linux

2.1 Preparing a Mass Storage Device

☑ Connect an HDD/SDD to your development Linux system, e.g., using a USB or SATA connection.

☑ Create a partition on the HDD, e.g., using command *cfdisk*.

☑ Format the partition with an ext3 file system with the following command (*/dev/sdc1* is the new partition):

```
$ mkfs.ext3 /dev/sdc1
```

☑ Mount the destination partition to */mnt* on your development Linux system.

```
$ mount /dev/sdc1 /mnt
```

For all further commands, the partition */dev/sdc1* mounted to */mnt* is used.

2.2 Configuring the IOP

2.2.1 Root File System

☑ Install the base system (mini) using *debootstrap* to the destination partition mounted on */mnt*. This will download a basic root file system for the IOP based on Ubuntu Trusty.

```
$ debootstrap --variant=minbase --arch=i386 trusty /mnt http://
de.archive.ubuntu.com/ubuntu/
```

Note: If you do not use Ubuntu Saucy on your development system (e.g., Ubuntu lucid) the script named *trusty* is not included in *debootstrap*.

Please download and install the newest version of *debootstrap* or use a different version of the root file system in the *debootstrap* command.

2.2.2 Additional Software Packages

The basic root file system does not include tools which are recommended for a running system. The following additional packages are just an example of some tools for a working system.

☑ Create a chroot of the new installed base system to install the tools directly on the target disk.

```
$ mount -o bind /dev /mnt/dev
$ mount -o bind /dev/pts /mnt/dev/pts
$ mount -t sysfs /sys /mnt/sys
$ mount -t proc /proc /mnt/proc
$ cp /proc/mounts /mnt/etc/mtab
$ cp /etc/resolv.conf /mnt/etc/resolv.conf
$ chroot /mnt
```

☑ Add the universe package source for additional tools to */etc/apt/source.lst* and update the repositories.

```
$ echo "deb http://de.archive.ubuntu.com/ubuntu/ trusty universe"
>> /etc/apt/sources.list
$ apt-get update
```

It is useful to add the following packages to the minimal root file system:

- *nano* (Editor)
- *net-tools* (ifconfig)
- *iputils-ping* (ping)
- *i2c-tools* (i2cdetect)
- *pciutils* (lspci)
- *unzip* (for MDIS)
- *build-essential* (make, gcc…)
- *openssh-client/openssh-server* (IOP)
- *debootstrapt* (download minimal base system)
- *minicom* (to communicate with the CPs via UART)

☑ Install the tools:

```
$ apt-get install nano net-tools i2c-tools pciutils unzip build-
essential openssh-client openssh-server debootstrap iputils-ping
minicom sudo
```

Note:The *openssh-server* package will display some errors like "Server cannot start". You can ignore these errors, because in the chroot installation environment it is not possible to start those servers.

☑ Exit the chroot:

```
$ exit
```

2.2.3 Building a Linux Kernel on the Development System

☑ Download the Linux kernel source (3.15.6) from kernel.org:

https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.15.6.tar.xz

☑ Unzip the sources. This will create a directory named *linux-3.15.6*.

```
$ tar xvfJ linux-3.15.6.tar.xz
```

☑ Navigate to the created folder and create a default kernel configuration:

```
$ cd linux-3.15.6
$ make defconfig
```

2.2.4 Configuring the Kernel

The default kernel configuration does not include all the drivers and components necessary for F75P. The following section describes which drivers are needed for the F75P IOP and CPs.

☑ Make sure the *ncurses* library is installed on the host system, as it is needed for *menuconfig*.

☑ You can install *ncurses* using the following command.

```
$ apt-get install libncurses5-dev
```

☑ Start the Linux Kernel Configuration menu in the kernel source directory.

```
$ make menuconfig
```

☑ Enable the following drivers (marked **bold**):

EXT3 Support

```
Filesystems --->
      Ext3 journalling file system support (*)
```

devtmpFS support

```
Device Drivers --->
   Generic Driver Options --->
      Maintain a devtmpfs filesystem to mount at /dev
```

I2C Support for Intel Atom and EG20T Chipset

```
Device Drivers --->
   I2C Support --->
      I2C device Interface (*)

      I2C Hardware Bus support --->
         Intel SCH SMBus 1.0 (*)
         Intel EG20T PCH/LAPIS Semicon IOH I2C (M)
```

GPIO Support

```
Device Drivers --->
   GPIO Support --->
      Intel SCH/TunnelCreek/Centeron GPIO (*)
      Intel EG20T PCH/LAPIS Semiconductor IOH GPIO (*)
      /sys/class/gpio… (sysfs interface) (*)
```

RTC Support

```
Device Drivers --->
   Real Time Clock --->
     Epson RX-8581 (*)
```

EG20T DMA Support

```
Device Drivers --->
   DMA Engine support --->
      Intel EG20T PCH/LAPIS Semicon IOH DMA (*)
```

NFS Server/Client Support

```
File systems --->
   Network File Systems --->
      NFS server support (*)
      NFS server support for NFS version 3 (*)
      NFS server support for NFS version 4 (*)
```

Serial Port Configuration

```
Device Drivers --->
   Character devices --->
      Serial drivers --->
         Number of 8250 serial ports to register at runtime ---> 10
```

Graphics Driver

```
Device Driver --->
   Graphic support --->
      Direct Rendering Manager --->
         Intel GMA5/600 KMS Framebuffer (*) --->
            Intel GMA600 support (*)
```

2.2.5 Modifying Drivers

There are some drivers in the kernel which will not work the way they are delivered. The kernel needs some patches to work correctly on F75P.

EG20T I2C Driver

The I2C driver for the Intel EG20T chipset assembled with the IOP needs to be modified because the driver for the I2C controller on the Intel Atom CPU cannot be loaded together with the EG20T chipset I2C driver.

☑ Open the source file in *linux-3.15.6/drivers/i2c/busses/i2c-eg20t.c* and replace the following function call in *pch_i2c_probe*:

```
// pch_adap->nr = i;
// ret = i2c_add_numbered_adapter(pch_adap);
ret = i2c_add_adapter(pch_adap);
```

Burst/Cache for Ethernet

The Ethernet buffers of each of the CPs are located in the FPGA rather than in the main memory of the CPU. To allow burst access to this buffer instead of single access to increase the transfer speed, the memory management kernel code needs to be modified to allow PCI Express burst access.

☑ Open the file *linux-3.15.6/arch/x86/mm/pat.c* and replace the *pat_x_mtrr_type* function with the following code.

```
static unsigned long pat_x_mtrr_type(u64 start, u64 end, unsigned
long req_type)
{
        /*
         * Look for MTRR hint to get the effective type in case
         * where PAT request is for WB.
         */
        if (req_type == _PAGE_CACHE_WB) {
                u8 mtrr_type;

                mtrr_type = mtrr_type_lookup(start, end);

                switch (mtrr_type) {
                        case MTRR_TYPE_WRBACK:
                        case MTRR_TYPE_WRTHROUGH:
                                return _PAGE_CACHE_WB;
                        default:
                                return _PAGE_CACHE_UC_MINUS;
                }
        }
        return req_type;
}
```

2.2.6 Building and Installing the Kernel

The kernel will be built on the development system and directly installed on the target disk.

☑ Build the kernel using option *-j* to start more threads during compilation depending on the development system CPU cores.

```
$ make -j2  // Compile kernel and modules
$ make modules_install INSTALL_MOD_PATH=/mnt/  // Install modules on target disk
```

☑ Install the kernel on the target disk.

```
$ cp System.map /mnt/boot
$ cp arch/x86/boot/bzImage /mnt/boot/vmlinuz-3.15.6
$ cp .config /mnt/boot/config-3.15.6
```

2.2.7 Installing the GRUB Boot Loader

Install the GRUB 2 boot loader which will load the kernel on the IOP:

☑ Create a chroot.

```
$ chroot /mnt
```

☑ Navigate to the *boot* directory.

```
$ cd /boot
```

☑ Create an initial *initrd*.

```
$ mkinitramfs -o initrd-3.15.6 3.15.6
```

☑ Download and install GRUB 2.

```
$ apt-get install grub2
```

GRUB will ask you where to install the boot loader. Select the disk of the connected target disk. (*/dev/sdc*) to install GRUB on.

☑ Open the GRUB configuration file.

```
$ nano /etc/default/grub
```

☑ To get debug outputs on boot, you have to change the following lines:

```
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_GFXMODE=text
```

☑ Update the new GRUB setting.

```
$ grub-mkconfig -o /boot/grub/grub.cfg
```

☑ After that you can exit the chroot environment.

```
$ exit
```

2.3 Installing MDIS

MEN's MDIS framework simplifies your system configuration for Linux. MEN provides a free MDIS System Package for Linux, which includes a tool to configure a system project step by step and many MEN device drivers. After configuring your MDIS project, you can build and install it.

2.3.1 Preparing for Installation

☑ Download the 13MD05-90 MDIS5 System Package for Linux from the MEN website:

www.men.de/products/13MD05-90.html

☑ Copy it to the home folder of the development machine.

☑ Unzip the *13md05-90.zip* file, e.g., to */home/men/mdis_sys*, e.g., using *unzip*.

```
$ unzip 13md05-90.zip
```

☑ Navigate to the MDIS installation directory and call the install script.

```
$ cd /home/men/mdis_sys
$ ./INSTALL
```

The script will copy all files to */opt/menlinux*.

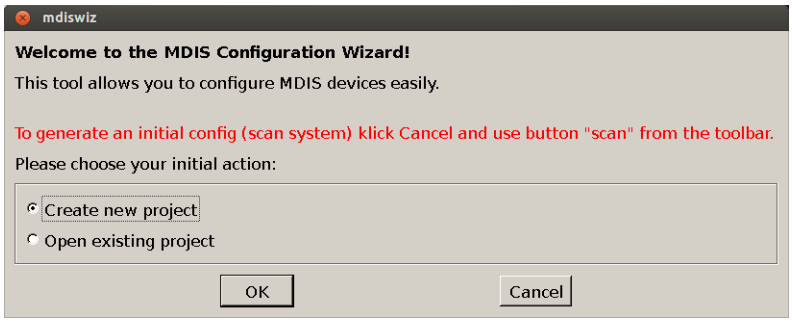
2.3.2 Configuring MDIS using MDIS Wizard

You can find details about the MDIS wizard in the *MDIS5 under Linux User Manual* available on the MEN website.

☑ Navigate to */opt/menlinux/BIN* and start the MDIS Configuration Wizard.

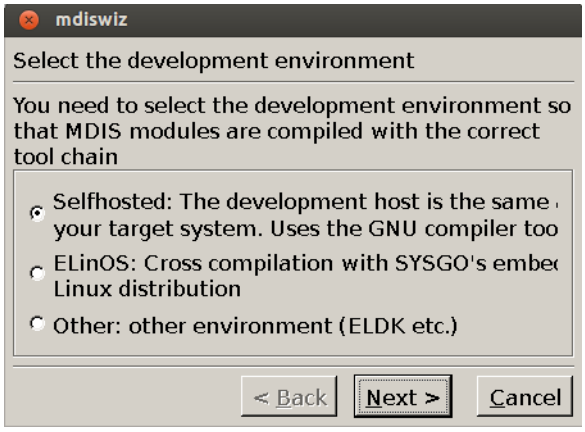
```
$ ./mdiswiz
```

The following dialog will open:

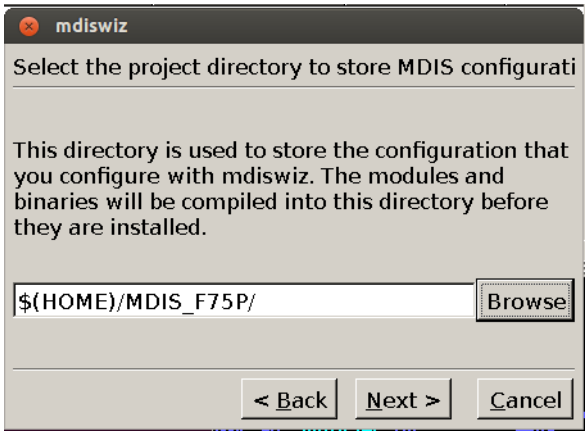

mdiswiz
Welcome to the MDIS Configuration Wizard!
This tool allows you to configure MDIS devices easily.
To generate an initial config (scan system) klick Cancel and use button "scan" from the toolbar.
Please choose your initial action:
Create new project
Open existing project
OK
Cancel

☑ Click OK to create a new MDIS project.

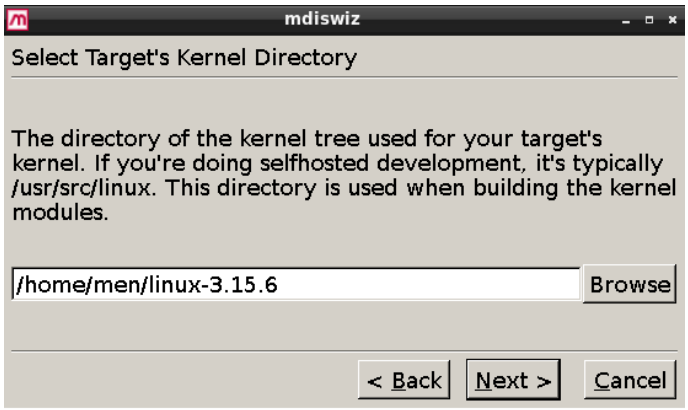
☑ Create a new selfhosted project.


mdiswiz
Select the development environment
You need to select the development environment so that MDIS modules are compiled with the correct tool chain
Selfhosted: The development host is the same your target system. Uses the GNU compiler too
ELinOS: Cross compilation with SYSGO's embe Linux distribution
Other: other environment (ELDK etc.)
< Back
Next >
Cancel

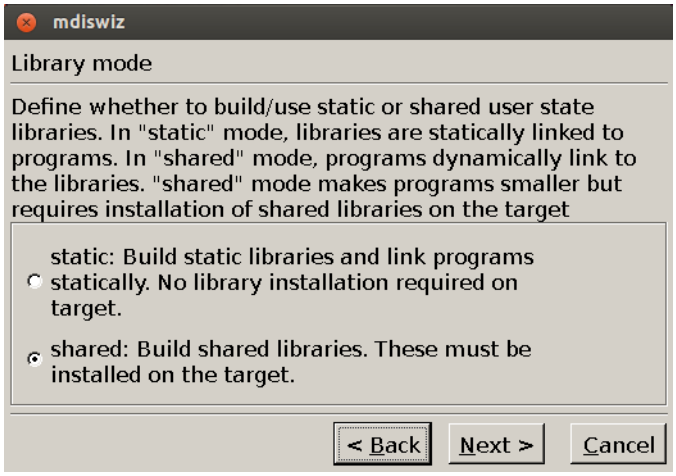
☑ Select a directory where to install the project to, e.g., create the project in the home directory of the development system.


mdiswiz
Select the project directory to store MDIS configurati
This directory is used to store the configuration that you configure with mdiswiz. The modules and binaries will be compiled into this directory before they are installed.
$(HOME)/MDIS_F75P/
Browse
< Back
Next >
Cancel

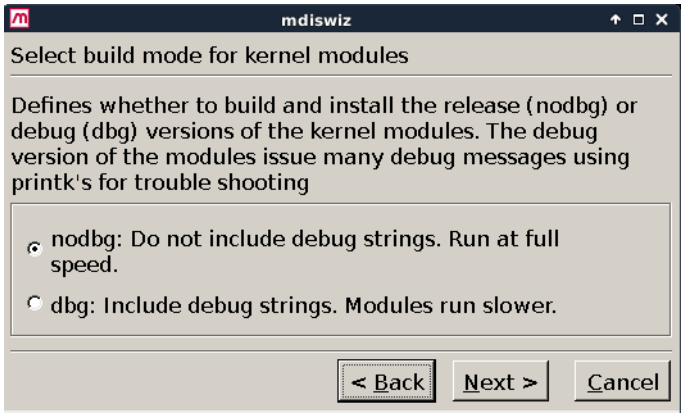
☑ Select the kernel sources for the kernel compiled before. The MDIS drivers will be compiled against this kernel.


mdiswiz
Select Target's Kernel Directory
The directory of the kernel tree used for your target's kernel. If you're doing selfhosted development, it's typically /usr/src/linux. This directory is used when building the kernel modules.
/home/men/linux-3.15.6
Browse
< Back
Next >
Cancel

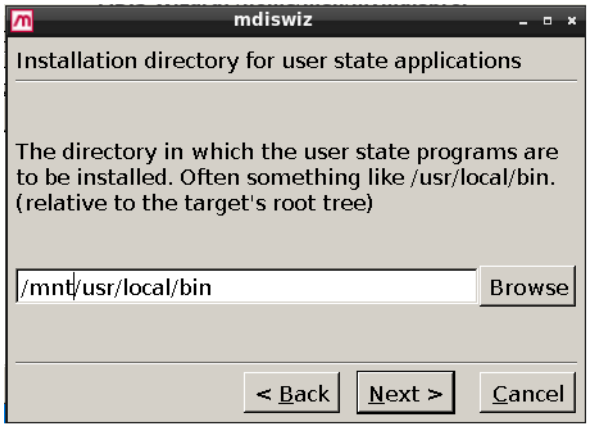
☑ Select shared or static libraries. Shared libraries will be copied directly to the target disk.

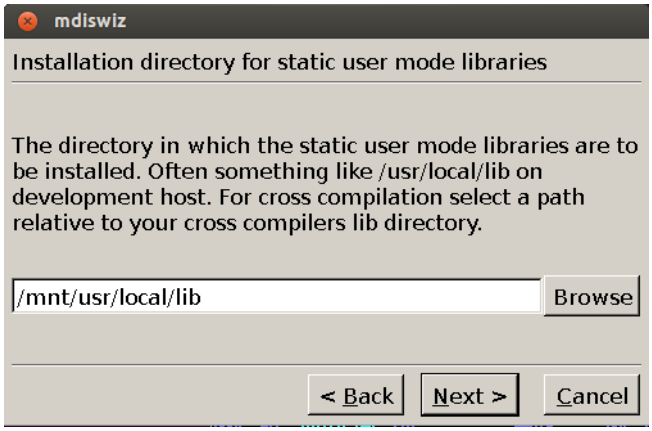

mdiswiz
Library mode
Define whether to build/use static or shared user state libraries. In "static" mode, libraries are statically linked to programs. In "shared" mode, programs dynamically link to the libraries. "shared" mode makes programs smaller but requires installation of shared libraries on the target
static: Build static libraries and link programs statically. No library installation required on target.
shared: Build shared libraries. These must be installed on the target.
< Back
Next >
Cancel

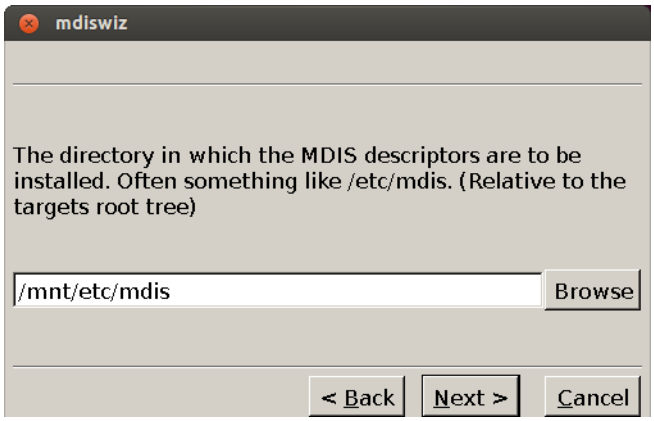
☑ Select if the *dbg* or *nodbg* drivers will be installed.

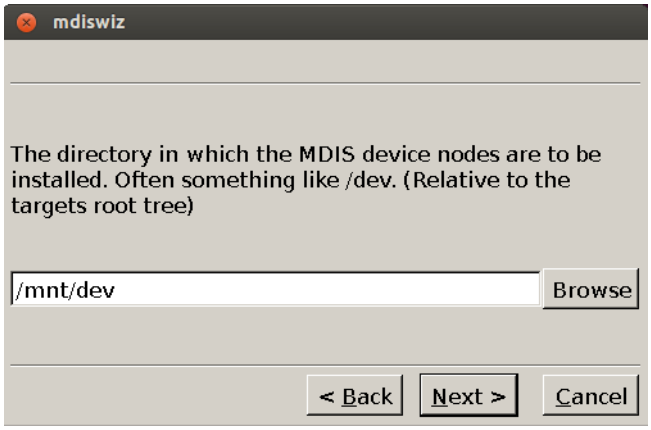

mdiswiz
Select build mode for kernel modules
Defines whether to build and install the release (nodbg) or debug (dbg) versions of the kernel modules. The debug version of the modules issue many debug messages using printk's for trouble shooting
nodbg: Do not include debug strings. Run at full speed.
dbg: Include debug strings. Modules run slower.
< Back
Next >
Cancel

☑ Select the target partition mounted on */mnt* to install the binaries (e.g., tools), libraries, MDIS descriptors and the MDIS device to.

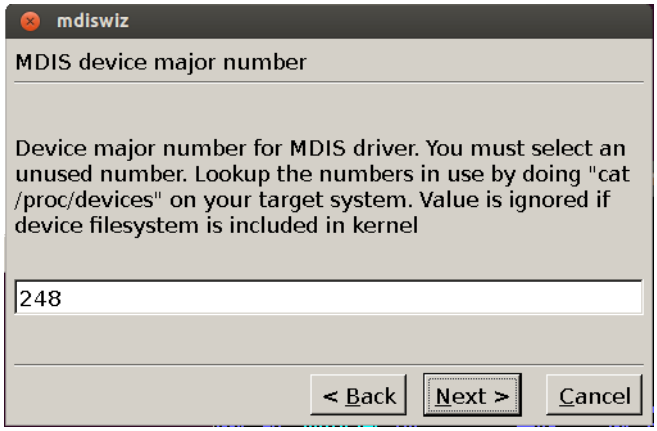

mdiswiz
Installation directory for user state applications
The directory in which the user state programs are to be installed. Often something like /usr/local/bin. (relative to the target's root tree)
/mnt/usr/local/bin
Browse
< Back
Next >
Cancel


mdiswiz
Installation directory for static user mode libraries
The directory in which the static user mode libraries are to be installed. Often something like /usr/local/lib on development host. For cross compilation select a path relative to your cross compilers lib directory.
/mnt/usr/local/lib
Browse
< Back
Next >
Cancel

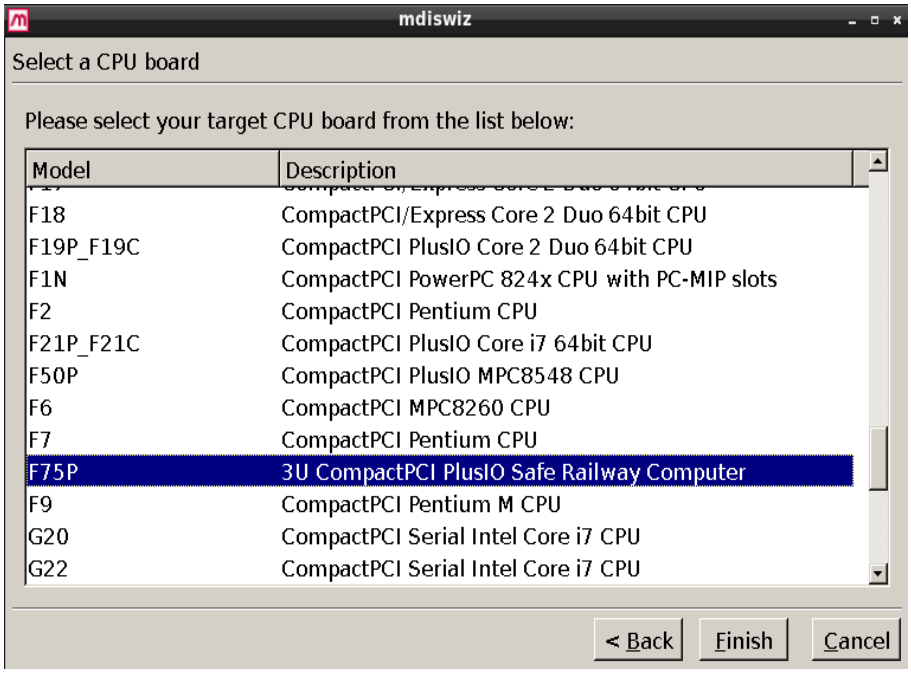

mdiswiz
The directory in which the MDIS descriptors are to be installed. Often something like /etc/mdis. (Relative to the targets root tree)
/mnt/etc/mdis
Browse
< Back
Next >
Cancel


mdiswiz
The directory in which the MDIS device nodes are to be installed. Often something like /dev. (Relative to the targets root tree)
/mnt/dev
Browse
< Back
Next >
Cancel

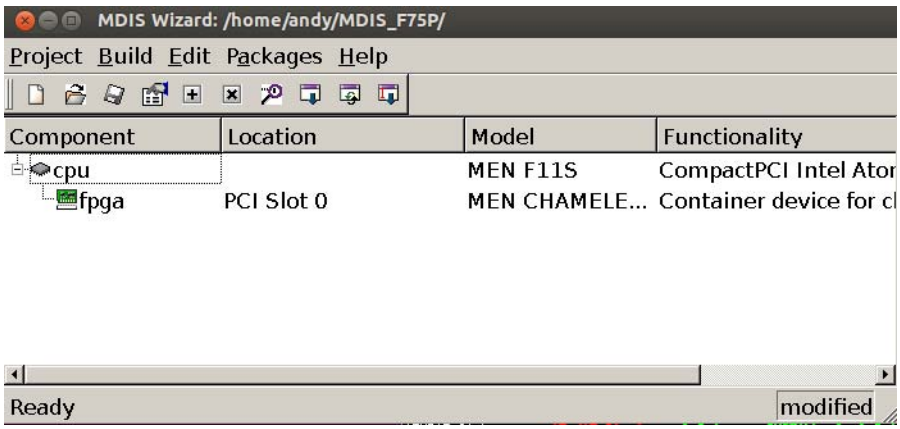
☑ Leave the default device major number for the MDIS device.


mdiswiz
MDIS device major number
Device major number for MDIS driver. You must select an unused number. Lookup the numbers in use by doing "cat /proc/devices" on your target system. Value is ignored if device filesystem is included in kernel
248
< Back
Next >
Cancel

☑ Select the F75P as CPU board for MDIS configuration.

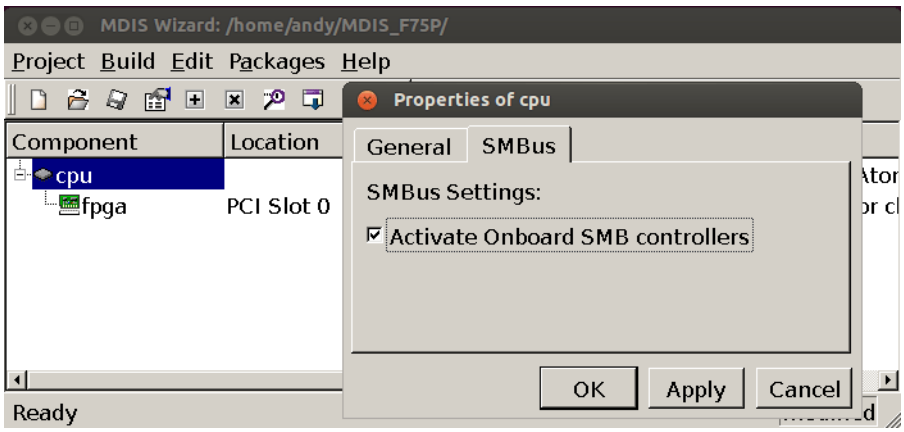


After that the MDIS Wizard will show a basic configuration with the CPU and the connected FPGA.

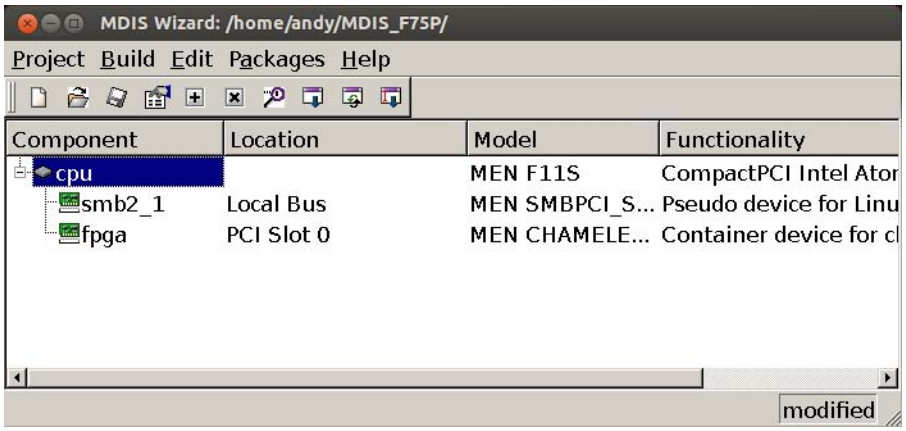


The next step will be to add the SMB driver/controller to the CPU.

☑ Right-click on *cpu* > *Properties* and navigate to SMBus to activate the onboard SMB controller.

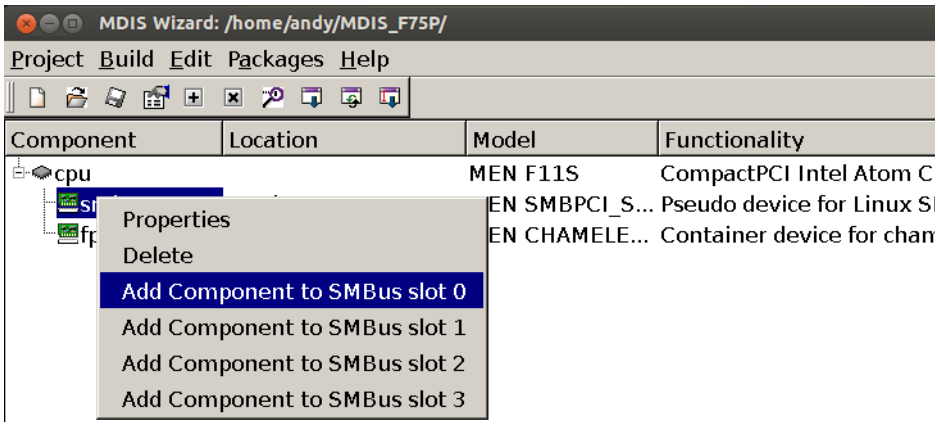


☑ Press OK to add the driver and leave the default settings. After that a new device named "smb2_1" will appear in the MDIS Wizard.

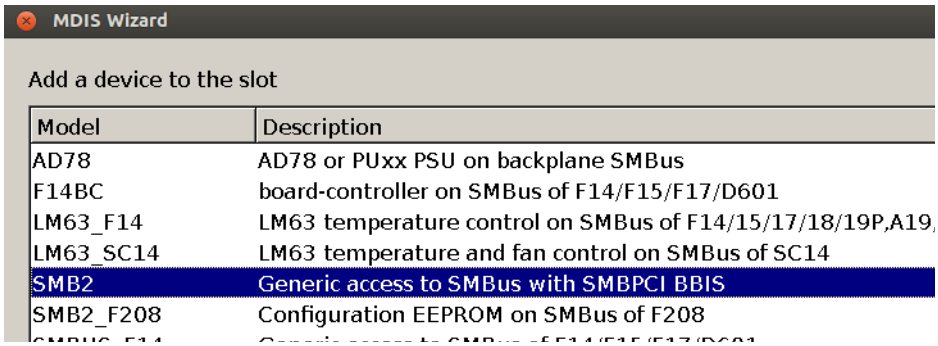


To get access to the SMBus devices, e.g., to the BMC (Board Management Controller), a new component named "Generic Access to SMBus" will be added.

☑ Right-click on the *smb2_1* device and select *Add Component to SMBus slot 0*.

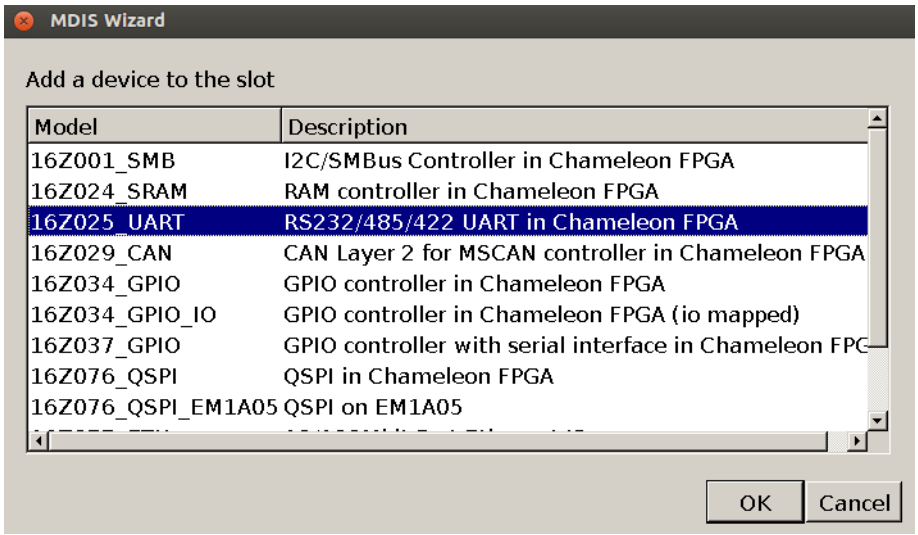


☑ A new window will appear with all of the available devices on the SMBus. For access to the BMC select the *SMB2 Generic access to SMBus with SMBPCI BBIS* device, and press "OK". Set the *SMB_BUSNBR* in the Descriptor section to "0" and press "OK" to quit.



☑ There are UART and Ethernet devices connected to the FPGA on F75P. To add these devices, right-click on *fpga* and select *Add Component to FPGA virtual slot X*.

☑ Select the *16Z025_UART* and press "OK".



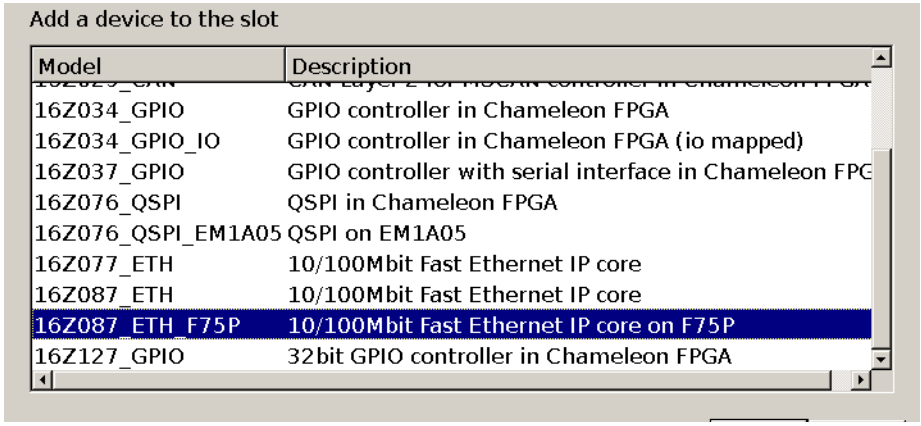
☑ Select an instance number, e.g., "1", and press "OK" to finish.

☑ Download the 13Z077-91 Ethernet driver package from the MEN website.

www.men.de/products/13Z077-91.html

☑ Install the driver using the MDIS Wizard: Select *Packages* > *Install Driver* in the MDIS Wizard menu and select file *13Z07791.zip*.

☑ Right-click on *fpga* again and add the *16Z087_ETH_F75P* Ethernet driver, with an instance number of "1", and press "OK" to finish.



☑ Save the project via *Project* > *Save* and exit the MDIS Wizard.

2.3.3 Building and Installing MDIS

After configuring your MDIS project, you can build and install it.

☑ Build the MDIS project using the following command in the project folder selected before (*$(HOME)/MDIS_F75P*).

```
$ make    //compile drivers
$ make install MODS_INSTALL_DIR=/mnt/lib/modules/<kernelversion>/
misc
          // install drivers to the target disc
```

☑ Change the root to your destination partition and update the modules.

```
$ chroot /mnt
$ depmod -a <kernelversion>
```

☑ The device node for the MDIS device will not be created automatically. Add the device node creation to */etc/rc.local*.

```
mknod /dev/mdis c 248 0
```

☑ Add the MEN MDIS kernel modules to */etc/modules*, so that the modules will be loaded on system boot.

```
#MEN Chameleon
men_lx_chameleon
#Frodo UART Driver
men_lx_frodo
#MEN SMB Driver
men_ll_smb2
```

2.3.4 Creating *initrd*

The IOP *initrd* must include the 16Z087_ETH_F75P Ethernet driver to be able to load the driver at system start-up.

Please make sure that the chroot is on */mnt*.

☑ Add the following modules to the configuration located in file */etc/initramfs-tools/modules*:

```
men_lx_chameleon
men_lx_z77_f75p mode=100FD,100FD,AUTO,AUTO,AUTO,AUTO phyadr=1,1,1,1,1,1
```

☑ Navigate to folder */boot/* and create *initrd*:

```
$ mkinitramfs -o <existing initrd filename> <kernelversion>
```

2.3.5 Configuring the Network

You need to configure static IP addresses for the interfaces from the IOP to CP1 and CP2.

☑ Add the following lines for *eth0* (interface to CP1) and *eth1* (interface to CP2) in */etc/network/interfaces*:

```
# Interface to CP1
auto eth0
iface eth0 inet static
      address 192.168.2.1
      netmask 255.255.255.0
# Interface to CP2
auto eth1
iface eth1 inet static
      address 192.168.3.1
      netmask 255.255.255.0
```

2.3.6 Creating a New User and Setting the Root Password

Logging in as a root user is not possible until a password is set. The new user "men" is optional.

☑ Do the following to create a user and set the root password:

```
$ adduser men
$ usermod -aG adm,plugdev,sudo men
$ passwd root
Enter new UNIX password: Type in your password here and press <Enter>
```

2.3.7 Installing a TFTP Server

☑ Install a TFTP server:

```
$ apt-get install atftpd
```

☑ Configure the TFTP server and set the path to the *tftpboot* folder. The configuration file is located in */etc/default/atftpd*. The *tftpboot* folder is used to provide the kernel, *initrd* and the root file system for each of the CPs. (The addition is marked **bold blue**.)

```
USE_INETD=false
OPTIONS="--tftpd-timeout 300 --retry-timeout 5 --mcast-port 1758 --
mcast-addr 239.239.239.0-255 --mcast-ttl 1 -maxthread 100 -
verbose=5 /var/lib/tftpboot
```

2.3.8 Installing a DHCP Server

☑ Install a DHCP server:

```
$ apt-get install isc-dhcp-server
```

☑ Configure the DHCP server with two subnetworks for CP1 and CP2. The configuration file is located in */etc/dhcp/dhcpd.conf*.
eth0 will be the interface from the IOP to CP1. *eth1* will be the interface from the IOP to CP2.

```
# DHCP for CP1
subnet 192.168.2.0 netmask 255.255.255.0 {
range 192.168.2.10 192.168.2.15;
interface eth0;
filename "/pxelinux.0";
}

# DHCP for CP2
subnet 192.168.3.0 netmask 255.255.255.0 {
interface eth1;
filename "/pxelinux.0";
range 192.168.3.10 192.168.3.15;
}
```

2.3.9 Installing an NFS Server

☑ Install an NFS (Network File System) server:

```
$ apt-get install nfs-kernel-server
```

Now you have to export the root file systems for each of the CPs. The root file system will be copied later to the path defined in the exports.

☑ Create the */etc/exports* file if it does not exist.

☑ Add the following lines in */etc/exports*:

```
/var/lib/tftpboot/CPX/cp1root  *(rw,no_all_squash,no_root_squash)
/var/lib/tftpboot/CPX/cp2root  *(rw,no_all_squash,no_root_squash)
```

2.3.10 Additional Modules Loading on Boot

Additional modules must be loaded on boot to provide I2C access.

☑ Add the following modules to */etc/modules*:

```
i2c-dev
i2c-eg20t
```

2.3.11 Instantiating the System RTC

The RTC device cannot enumerate through the kernel and must be initiated manually.

☑ Add the following line to */etc/rc.local* to add the RTC device on Bus 0 address 0x32.

```
echo rx8581 0x32 > /sys/bus/i2c/devices/i2c-0/new_device
```

2.4 Configuring CP1 and CP2

In this example configuration, some of the following steps are done only for CP1. The respective files are later duplicated and/or adapted. This means that the two Control Processors are configured to be identical.

2.4.1 CP Root File System

Each of the CPs needs a root file system provided by the IOP which will be loaded through the NFS (Network File System).

☑ Create a folder in */var/lib/tftpboot/CPX* for the root file system of CP1.

```
$ mkdir -p /var/lib/tftpboot/CPX
$ mkdir /var/lib/tftpboot/CPX/cp1root
```

☑ Download a minimal root file system using *debootstrap* to folder *cp1root*.

```
$ debootstrap --variant=minbase --arch=i386 trusty /var/lib/
tftpboot/CPX/cp1root http://de.archive.ubuntu.com/ubuntu/
```

☑ Copy the modules from the compiled IOP kernel to the root file system of CP1.

```
$ mkdir /var/lib/tftpboot/CPX/cp1root/lib/modules
$ cp -rf /lib/modules/<kernelversion> /var/lib/tftpboot/CPX/
cp1root/lib/modules
```

☑ Copy the compiled IOP kernel to the *tftpd* folder and change the permissions.

```
$ cp /boot/vmlinuz-<kernelversion> /var/lib/tftpboot/CPX/vmlinuz
$ chmod 770 /var/lib/tftpboot/CPX/vmlinuz
```

2.4.2 Additional Software for the CP

☑ Create a chroot in the *cp1root* directory to install additional software for CP1.

```
$ chroot /var/lib/tftpboot/CPX/cp1root
```

☑ Add the universe package source for additional tools to */etc/apt/source.lst* and update the repositories.

```
$ echo "deb http://de.archive.ubuntu.com/ubuntu/ trusty universe"
>> /etc/apt/sources.list
$ apt-get update
```

☑ Install the tools:

```
$ apt-get install nano net-tools i2c-tools pciutils openssh-server
iputils-ping sudo
```

☑ Exit the chroot:

```
$ exit
```

2.4.3 Creating Configuration Files for PXE Boot

Now you need to create two configuration files for PXE boot, for CP1 and CP2, in folder */var/lib/tftpboot/CPX*.

You need to create a file for each CP. The file name depends on the IP address given for the CP, e.g., `C0A802` is a hexadecimal number meaning a 192.168.2.0 network.

☑ Create a configuration file for CP1 named *C0A802* and insert the following two lines:

```
1 Label linux
2       DEFAULT CPX/vmlinuz ip=:::::eth0:dhcp netboot=nfs
nfsroot=192.168.2.1:/var/lib/tftpboot/CPX/cp1root rw showopts
initrd=CPX/initrd-CP1.img acpi=off
```

☑ Create a configuration file for CP2 named *C0A803* and insert the following two lines:

```
1 Label linux
2       DEFAULT CPX/vmlinuz ip=:::::eth0:dhcp netboot=nfs
nfsroot=192.168.3.1:/var/lib/tftpboot/CPX/cp2root rw showopts
initrd=CPX/initrd-CP2.img acpi=off
```

PXE Linux needs a file named *pxelinux.cfg* in the root of the *tftpboot* folder.

☑ Create a symlink to CPX from *pxelinux.cfg* so the TFTP server can find the configuration files.
Navigate to directory */var/lib/tftpboot* and create the symlink:

```
$ ln -s CPX pxelinux.cfg
```

2.4.4 Downloading the PXE Linux Boot File

☑ Navigate to */var/lib/tftpboot/* and download the *pxelinux.0* file from the Ubuntu server.

```
$ wget http://de.archive.ubuntu.com/ubuntu/dists/trusty/main/
installer-i386/current/images/netboot/pxelinux.0
```

2.4.5 Creating *initrd* for Each of the CPs

☑ Open the *initramfs* configuration file located in */var/lib/tftpboot/CPX/cp1root/etc/initramfs-tools/initramfs.conf* and set the following options for NFS net-boot.

```
MODULES=netboot
BOOT=nfs
```

☑ Open the *initramfs modules* file located in */var/lib/tftpboot/CPX/cp1root/etc/initramfs-tools/modules* and add the following modules to be loaded on boot.

```
#MEN Chameleon (FPGA)
men_lx_chameleon
men_chameleon_io
#MEN 16Z087_ETH Ethernet Driver
men_lx_z77_f75p mode=100FD,100FD phyadr=1,1 nodma=1,1
men_oss
#MEN Frodo UART Driver
men_lx_frodo
```

Now, you need to create the *initrd* image for each of the CPs.

☑ Navigate to */var/lib/tftpboot/CPX*.

☑ Create the newly configured *initrd* image for CP1.

```
$ mkinitramfs -d cp1root/etc/initramfs-tools -o initrd-CP1.img
<kernelversion>
```

☑ Duplicate the *initrd* image for CP2.

```
$ cp initrd-CP1.img initrd-CP2.img
```

☑ Duplicate the root file system for CP2.

```
$ cp -rf cp1root cp2root
```

2.4.6 Setting the Host Name for Each CP

☑ Set a host name for each of the CPs in the root file system.

```
$ echo "CP1" > /var/lib/tftpboot/CPX/cp1root/etc/hostname
$ echo "CP2" > /var/lib/tftpboot/CPX/cp2root/etc/hostname
```

2.4.7 Setting the Password for Root in CP1 and CP2

To get root access, you need to change the password for each of the CPs. Also, add a user account named, e.g., 'men'.

☑ Change the password for CP1.

```
$ chroot /var/lib/tftpboot/CPX/cp1root
$ passwd root
Enter new UNIX password: Type in your CP1 password here and press <Enter>
```

☑ Add a user account for CP1.

```
$ adduser men
Enter new UNIX password: Type in your user password here and press <Enter>
$ usermod -aG adm,plugdev,sudo men
$ exit
```

☑ Change the password for CP2.

```
$ chroot /var/lib/tftpboot/CPX/cp2root
$ passwd root
Enter new UNIX password: Type in your CP2 password here and press <Enter>
```

☑ Add a user account for CP2.

```
$ adduser men
Enter new UNIX password: Type in your user password here and press <Enter>
$ usermod -aG adm,plugdev,sudo men
$ exit    chroot of cp2root
$ exit    chroot of /mnt
```

2.4.8 Configuring the Watchdog Trigger using a Script

Each of the CPs' software has to configure and trigger the watchdog. There are GPIOs connected to the CPU to configure and trigger the watchdog. Theses are supported by a driver named *gpio_sch*. This driver exports the GPIO features to the *sysfs* file system. A bash script can be used to do this.

Please note that this is just an example to get the OS running. The final application software running on each of the CPs has to trigger the watchdog.

☑ Create the following script in each CP root file system and add it to the respective *rc.local* file, so that the watchdog is triggered at start-up of the CPs' Linux OS.

```
#/!/bin/bash

################################################
# Export GPIO to access them from User Space
################################################
echo 2 > /sys/class/gpio/export # GPIO 2 CP_WD_TRIG
echo 9 > /sys/class/gpio/export # GPIO SUS4 CP_WD_TOUT_L
echo 10 > /sys/class/gpio/export # GPIO SUS5 CP_WD_TOUT_U2
echo 11 > /sys/class/gpio/export # GPIO SUS6 CP_WD_TOUT_U1
echo 12 > /sys/class/gpio/export # GPIO SUS7 CO_WD_TOUT_U0

########################################
# Configure GPIO as Outputs
########################################
echo out > /sys/class/gpio/gpio2/direction
echo out > /sys/class/gpio/gpio9/direction
echo out > /sys/class/gpio/gpio10/direction
echo out > /sys/class/gpio/gpio11/direction
echo out > /sys/class/gpio/gpio12/direction

###################################
# Set Watchdog timout to 2560ms
###################################
echo 1 > /sys/class/gpio/gpio10/value
echo 1 > /sys/class/gpio/gpio11/value
echo 1 > /sys/class/gpio/gpio12/value
echo 0 > /sys/class/gpio/gpio9/value

################################################
# Trigger watchdog once to set the timout value
################################################
echo 1 > /sys/class/gpio/gpio2/value
sleep 0.2
echo 0 > /sys/class/gpio/gpio2/value

#################################
# Start trigger the Watchdog
#################################
while true; do
      echo "Trigger WD"
      echo 1 > /sys/class/gpio/gpio2/value
      sleep 0.5
      echo 0 > /sys/class/gpio/gpio2/value
      sleep 0.5
done
```

2.5 Using the Installation on F75P

The installation is finished now.

☑ Disconnect the mass storage device containing the installation from your development system and connect it to the F75P.

☑ Power up F75P.

The IOP will boot Linux and will provide the files needed for each of the CPs to be able to boot.

After the system has started, the CPs are accessible on IOP through SSH:

- CP1 is available in an IP address range of 192.168.2.10 to 192.168.2.15.
- CP2 is available in an IP address range of 192.168.3.11 to 192.168.2.15.

3 Using F75P Functions under Linux

This chapter gives hints on how to use some dedicated functions of F75P, especially where specific software support is already in place. It is **not** a complete functional description.

Tools available within the MDIS framework were built during the MDIS build process.

3.1 Graphical User Interface (GUI)

If you want to install a Graphical User Interface (GUI), connect the SATA mass storage device again to the development machine.

☑ Mount the disk and create a chroot.

```
$ mount /dev/sdc1 /mnt
$ chroot /mnt
```

☑ This example shows how to install the LXDE desktop environment using the LXDM display manager as login manager.

```
$ apt-get install lxde
```

3.2 Real-Time Clock (RTC)

The system RTC is connected to I2C Bus 0 device address `0x32` and will be instantiated manually by the */etc/rc.local* script described in Chapter 2.3.11 Instantiating the System RTC on page 27.

There is a native Linux tool named *hwclock* to access this device using device node */dev/rtc1* which will be created by the RX8581 RTC driver.

Reading the RTC Time

```
$ hwclock -r -f /dev/rtc1
```

Setting the RTC Time

```
$ hwclock --set --date "07/29/2014 17:10:10" -f /dev/rtc1 -localtime
```

Setting the RTC Time to System Time

Please note that the RTC additionally implemented in the F75P is considered the system RTC. The real-time clocks integrated in each Intel Atom E6xx processor are **not automatically synchronized with the system RTC**. The IOP's processor clock may be used by standard software modules. The *hwclock* tool provides one way of synchronizing the F75P system RTC and processor RTCs.

```
$ hwclock --hctosys -f /dev/rtc1
```

For more information on the RTC implementation on F75P, please refer to the *F75P User Manual*, Chapter 3.7 Real-Time Clock (RTC).

3.3 Board Management Controller (BMC)

There is a tool named *smb2_bmc* to access the board management controller. This provides access, e.g., to the event log.

You can find the source code of this tool in the MDIS directory that was created during the installation of Linux as described in Chapter 2 Installing Linux in */opt/menlinux/MDIS_LL/SMB2/TOOLS/SMB2_BMC*.

Displaying a Function Overview

The tool includes a complete help function that displays a command list and also includes examples of how to call the tool.

```
$ smb2_bmc

Usage:     smb2_bmc <dev-name> <smb-addr> <cmd> [<opts>]

Function:  Tool to control SMBus-BMCs (e.g. on F75P)
           via BMC-API commands
Options:
  dev-name       : device name e.g. smb2_1
  smb-addr       : smb device address (default: 0x9C)
  cmd            : command to execute

Command List:

  --- FS_BMC_MANAGEMENT ---
  gfr            : get firmware revision
  ghwb           : get hw board
  shwb <boardid> : set hw board - default: F75P (ID:0x0010)
  feat           : features

  --- FS_WATCHDOG ---
  wdo            : wdog on
  wdf            : wdog off
  wdt     <opts> : wdog trigger time - in milliseconds [30000]
  wdtg           : wdog time get

....

Calling examples:

 - get firmware revision:
     smb2_bmc smb2_1 0x9c gfr

 - software reset:
     smb2_bmc smb2_1 0x9c swr

(c)Copyright 2013 by MEN Mikro Elektronik GmbH
$Id: smb2_bmc.c,v 1.1 2014/01/07 17:38:02 MRoth Exp $
```

Reading out the Firmware Revision

```
$ smb2_bmc smb2_2 0x9c gfr

firmware revision: 1.1.0
build number: 2
verified flag: 0x0
```

3.4 Board Information EEPROM

There is an MDIS tool named *smb2_boardident* to read out the board information EEPROM. You can find the source code of this tool in the MDIS directory that was created during the installation of Linux as described in Chapter 2 Installing Linux in */opt/menlinux/MDIS_LL/SMB2/TOOLS/SMB2_BOARDIDENT*.

Reading the Board Information EEPROM

```
$ smb2_boardident smb2_2
```

Example output:

```
============================
===   SMB2_BOARDIDENT   ===
============================
(c)Copyright 2009 by MEN Mikro Elektronik GmbH
$Id: smb2_boardident.c,v 1.3 2010/02/11 15:08:01 MRoth Exp $

Accessing smb2_2: smbAddr 0xae

DUMP EEPROM:

 EEPROD-ID    = 0xE
 Serial#      = 00000001
 Revision     = 02.01.00
 HW-Name      = F75P00
 Prod. Date   = 2014-05-05
 Rep. Date    = FFFF-FF-FF
```

4 Demo Application

This chapter describes the concept, installation and functions of a Linux demo application that is available from MEN for the F75P.

4.1 About the Application

The F75P demo application for Linux is a traffic light simulator that demonstrates the interaction between the three F75P processors CP1, CP2 and IOP. It consists of three independent tools that run on the three processors.

The demo application is available for download on the F75P product pages, under "Downloads":
www.men.de/products/02F075P.html

4.1.1 General Hardware and Software Requirements

To be able to reasonably work with the F75P demo application, we recommend that your target system is completely configured and running. This includes your hardware components, but also the software (Linux OS with the necessary packages and drivers, see Chapter 2 Installing Linux and Chapter 4.2 Installing the Demo Application on page 37).

4.2 Installing the Demo Application

In order to use the demo application, you need running operating system images for all three F75P processors. Chapter 2 Installing Linux gives you all instructions to do this.

To install the demo application, do the following:

☑ Download the demo application (article no. 13Y026-90) from the MEN website.

The demo application is available for download on the F75P product pages, under "Downloads":
www.men.de/products/02F075P.html

☑ Install an X-Server and a window manager such as Gnome.

☑ Install the Python 3.x package.

☑ Install the Tkinter graphical user interface package (*python3-tk*).

☑ Unzip the *F75P_Demo* package to a folder on your target system (e.g., to */home/user/*).

☑ Call *make* to create the *f75p_demo_cp* binary.

```
$ make
```

☑ Copy the *f75p_demo_cp* binary to the CP1 root folder.

```
$ cp f75p_demo_cp /var/lib/tftpboot/CPX/cp1root/root
```

☑ Copy the *f75p_demo_cp.py* script to the CP2 root folder.

```
$ cp f75p_demo_cp.py /var/lib/tftpboot/CPX/cp2root/root
```

☑ Add the tools to the *rc.local* files (autostart) of the two CP images.

For CP1:

```
/var/lib/tftpboot/CPX/cp1root/etc/rc.local
./f75p_demo_cp 192.168.3.1 &
```

For CP2:

```
/var/lib/tftpboot/CPX/cp2root/etc/rc.local
python2.7 f75p_demo_cp.py 192.168.2.1 &
```

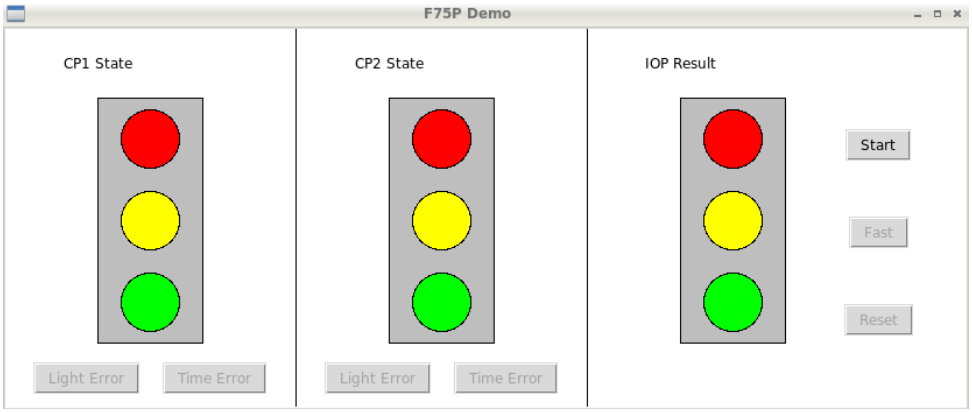
4.3 Using the Demo Application

4.3.1 Start

You can start the *F75P_Demo* application in the IOP Linux by calling the *f75p_demo_iop.py* script:

```
$ python3.3 f75p_demo_iop.py
```

You will see the following graphical user interface:



Press the *Start* button on the right to start the traffic light simulator.

4.3.2 General Behavior of the Application

Without any user interference, the traffic lights behave just like any real-life traffic lights. The red and green traffic light states will last 12 seconds each, and the yellow (orange/amber) light states will last 4 seconds.

The IOP application acts as a voter and visualizes the states of the CPs. If both CPs have the same state, the traffic light of the IOP changes according to the state of the CPs. If the results of the two CPs don't match (as provoked by the error functions, see below), the IOP signals an error.

4.3.3 Functions

The buttons of the user interface simulate error, signal and reset injections.

Fast

The *Fast* button simulates pedestrians that press the traffic light button to pass the street. The CPs will get the signal to switch faster to red or green and the state machine of the CPs will be adjusted.

Time Error

The *Time Error* buttons inject a "sleep" value to the respective CP. If the CP determines that the *Time Error* button was pressed, the CP will send its current state later than normal. The IOP recognizes the time difference between the two CPs and will start flashing the yellow (orange/amber) light.

The flashing yellow light simulates the safe state of the F75P.

Light Error

The *Light Error* buttons inject a state machine error to the CPs. The CPs will then send back an error state to the IOP. The error state of the CP will be visualized by "all lights on", i.e. all three lights are active on the faulty CP.

Similarly to the Time Error, the IOP will start flashing the yellow (orange/amber) light to simulate the safe state of the F75P.

Reset

The *Reset* button is used to reset the CP states to the starting condition, which is a red light. The reset signal also restarts the IOP voter.

5 Appendix

5.1 Literature and Downloads

5.1.1 CPU Board and BIOS

F75P product pages:
www.men.de/products/02F075P.html
You can find documents and downloads under "Downloads".

Some documents may only be available on request or within a bundle.
Please contact MEN sales for more information.

5.1.2 Ubuntu Linux

- **Ubuntu Linux pages**
  www.ubuntu.com
- **Ubuntu support**
  www.ubuntu.com/support

5.1.3 Embedded Linux

- **Building Embedded Linux Systems - 2nd Edition**
  Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, Philippe Gerum; 2008; O'Reilly Media; ISBN 978-0-596-52968-0
- **Programming Embedded Linux**
  Michael Opdenacker, Oron Peled; 2006; Codefidence Ltd.
  Free download: www.scribd.com/doc/3009867/Programming-Embedded-Linux

5.1.4 Linux Kernel Configuration

Linux Kernel in a Nutshell
Greg Kroah-Hartman; 2007; O'Reilly Media; ISBN 978-0-596-10079-7
Free download: www.kroah.com/lkn/

5.1.5 Linux Device Drivers

- **Linux Device Drivers - 3rd Edition**
  Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman; 2005; O'Reilly Media; ISBN 0-596-00590-3
  Free download: http://lwn.net/Kernel/LDD3/
- **Linux Kernel Development - 2nd Edition**
  Robert Love; 2005; Novell Press

5.1.6 Linux System Programming

- **Linux System Programming**
  Robert Love; 2007; O'Reilly Media; ISBN 978-0-596-00958-8
- **Advanced Linux Programming**
  Mark Mitchell, Jeffrey Oldham, Alex Samuel; 2001; New Riders Publishing; ISBN 0-7357-1043-0
  Free download: www.advancedlinuxprogramming.com/alp-folder

5.1.7 General

- **Linux in a Nutshell - 5th Edition**
  Ellen Siever, Aaron Weber, Stephen Figgins, Robert Love, Arnold Robbins;
  2005; O'Reilly Media; ISBN 978-0-596-00930-4
- **Linux from Scratch**
  Gerard Beekmans; 2008
  Free download: www.linuxfromscratch.org